Title

stata.com

mgarch vcc — Varying conditional correlation multivariate GARCH models

Description

`mgarch vcc` estimates the parameters of varying conditional correlation (VCC) multivariate generalized autoregressive conditionally heteroskedastic (MGARCH) models in which the conditional variances are modeled as univariate generalized autoregressive conditionally heteroskedastic (GARCH) models and the conditional covariances are modeled as nonlinear functions of the conditional variances. The conditional correlation parameters that weight the nonlinear combinations of the conditional variance follow the GARCH-like process specified in Tse and Tsui (2002).

The VCC MGARCH model is about as flexible as the closely related dynamic conditional correlation MGARCH model (see [TS] **mgarch dcc**), more flexible than the conditional correlation MGARCH model (see [TS] **mgarch ccc**), and more parsimonious than the diagonal vech model (see [TS] **mgarch dvech**).

Quick start

Fit varying conditional correlation multivariate GARCH with first- and second-order ARCH components for dependent variables `y1` and `y2` using `tsset` data

```
mgarch vcc (y1 y2), arch(1 2)
```

Add regressors `x1` and `x2` and first-order GARCH component

```
mgarch vcc (y1 y2 = x1 x2), arch(1 2) garch(1)
```

Add `z1` to the model for the conditional heteroskedasticity

```
mgarch vcc (y1 y2 = x1 x2), arch(1 2) garch(1) het(z1)
```

Menu

Statistics > Multivariate time series > Multivariate GARCH

Syntax

```
mgarch vcc eq [eq ... eq] [if] [in] [, options]
```

where each *eq* has the form

```
(depvars = [indepvars] [, eqoptions])
```

options	Description
Model	
`arch(`*numlist*`)`	ARCH terms for all equations
`garch(`*numlist*`)`	GARCH terms for all equations
`het(`*varlist*`)`	include *varlist* in the specification of the conditional variance for all equations
`distribution(`*dist* `[#])`	use *dist* distribution for errors [may be `gaussian` (synonym `normal`) or `t`; default is `gaussian`]
`constraints(`*numlist*`)`	apply linear constraints
SE/Robust	
`vce(`*vcetype*`)`	*vcetype* may be `oim` or `robust`
Reporting	
`level(#)`	set confidence level; default is `level(95)`
`nocnsreport`	do not display constraints
display_options	control columns and column formats, row spacing, line width, display of omitted variables and base and empty cells, and factor-variable labeling
Maximization	
maximize_options	control the maximization process; seldom used
`from(`*matname*`)`	initial values for the coefficients; seldom used
`coeflegend`	display legend instead of statistics

eqoptions	Description
Model	
`noconstant`	suppress constant term in the mean equation
`arch(`*numlist*`)`	ARCH terms
`garch(`*numlist*`)`	GARCH terms
`het(`*varlist*`)`	include *varlist* in the specification of the conditional variance

You must `tsset` your data before using `mgarch vcc`; see [TS] **tsset**.
indepvars and *varlist* may contain factor variables; see [U] **11.4.3 Factor variables**.
depvars, *indepvars*, and *varlist* may contain time-series operators; see [U] **11.4.4 Time-series varlists**.
`by`, `fp`, `rolling`, and `statsby` are allowed; see [U] **11.1.10 Prefix commands**.
`coeflegend` does not appear in the dialog box.
See [U] **20 Estimation and postestimation commands** for more capabilities of estimation commands.

Options

Model

`arch(`*numlist*`)` specifies the ARCH terms for all equations in the model. By default, no ARCH terms are specified.

`garch(`*numlist*`)` specifies the GARCH terms for all equations in the model. By default, no GARCH terms are specified.

`het(`*varlist*`)` specifies that *varlist* be included in the model in the specification of the conditional variance for all equations. This varlist enters the variance specification collectively as multiplicative heteroskedasticity.

`distribution(`*dist* [*#*]`)` specifies the assumed distribution for the errors. *dist* may be `gaussian`, `normal`, or `t`.

`gaussian` and `normal` are synonyms; each causes `mgarch vcc` to assume that the errors come from a multivariate normal distribution. *#* may not be specified with either of them.

`t` causes `mgarch vcc` to assume that the errors follow a multivariate Student t distribution, and the degree-of-freedom parameter is estimated along with the other parameters of the model. If `distribution(t #)` is specified, then `mgarch vcc` uses a multivariate Student t distribution with *#* degrees of freedom. *#* must be greater than 2.

`constraints(`*numlist*`)` specifies linear constraints to apply to the parameter estimates.

SE/Robust

`vce(`*vcetype*`)` specifies the estimator for the variance–covariance matrix of the estimator.

`vce(oim)`, the default, specifies to use the observed information matrix (OIM) estimator.

`vce(robust)` specifies to use the Huber/White/sandwich estimator.

Reporting

`level(#)`; see [R] **estimation options**.

`nocnsreport`; see [R] **estimation options**.

display_options: `noci`, `nopvalues`, `noomitted`, `vsquish`, `noemptycells`, `baselevels`, `allbaselevels`, `nofvlabel`, `fvwrap(#)`, `fvwrapon(`*style*`)`, `cformat(%`*fmt*`)`, `pformat(%`*fmt*`)`, `sformat(%`*fmt*`)`, and `nolstretch`; see [R] **estimation options**.

Maximization

maximize_options: `difficult`, `technique(`*algorithm_spec*`)`, `iterate(#)`, [`no`]`log`, `trace`, `gradient`, `showstep`, `hessian`, `showtolerance`, `tolerance(#)`, `ltolerance(#)`, `nrtolerance(#)`, `nonrtolerance`, and `from(`*matname*`)`; see [R] **maximize** for all options except `from()`, and see below for information on `from()`. These options are seldom used.

`from(`*matname*`)` specifies initial values for the coefficients. `from(b0)` causes `mgarch vcc` to begin the optimization algorithm with the values in `b0`. `b0` must be a row vector, and the number of columns must equal the number of parameters in the model.

The following option is available with `mgarch vcc` but is not shown in the dialog box:

`coeflegend`; see [R] **estimation options**.

Eqoptions

`noconstant` suppresses the constant term in the mean equation.

`arch(`*numlist*`)` specifies the ARCH terms in the equation. By default, no ARCH terms are specified. This option may not be specified with model-level `arch()`.

`garch(`*numlist*`)` specifies the GARCH terms in the equation. By default, no GARCH terms are specified. This option may not be specified with model-level `garch()`.

`het(`*varlist*`)` specifies that *varlist* be included in the specification of the conditional variance. This varlist enters the variance specification collectively as multiplicative heteroskedasticity. This option may not be specified with model-level `het()`.

Remarks and examples

stata.com

We assume that you have already read [TS] **mgarch**, which provides an introduction to MGARCH models and the methods implemented in `mgarch vcc`.

MGARCH models are dynamic multivariate regression models in which the conditional variances and covariances of the errors follow an autoregressive-moving-average structure. The VCC MGARCH model uses a nonlinear combination of univariate GARCH models with time-varying cross-equation weights to model the conditional covariance matrix of the errors.

As discussed in [TS] **mgarch**, MGARCH models differ in the parsimony and flexibility of their specifications for a time-varying conditional covariance matrix of the disturbances, denoted by $\mathbf{H}_t$. In the conditional correlation family of MGARCH models, the diagonal elements of $\mathbf{H}_t$ are modeled as univariate GARCH models, whereas the off-diagonal elements are modeled as nonlinear functions of the diagonal terms. In the VCC MGARCH model,

$$h_{ij,t} = \rho_{ij,t}\sqrt{h_{ii,t}h_{jj,t}}$$

where the diagonal elements $h_{ii,t}$ and $h_{jj,t}$ follow univariate GARCH processes and $\rho_{ij,t}$ follows the dynamic process specified in Tse and Tsui (2002) and discussed below.

Because the $\rho_{ij,t}$ varies with time, this model is known as the VCC GARCH model.

❑ Technical note

The VCC GARCH model proposed by Tse and Tsui (2002) can be written as

$$\begin{aligned}
\mathbf{y}_t &= \mathbf{C}\mathbf{x}_t + \boldsymbol{\epsilon}_t \\
\boldsymbol{\epsilon}_t &= \mathbf{H}_t^{1/2}\boldsymbol{\nu}_t \\
\mathbf{H}_t &= \mathbf{D}_t^{1/2}\mathbf{R}_t\mathbf{D}_t^{1/2} \\
\mathbf{R}_t &= (1-\lambda_1-\lambda_2)\mathbf{R} + \lambda_1\boldsymbol{\Psi}_{t-1} + \lambda_2\mathbf{R}_{t-1}
\end{aligned} \tag{1}$$

where

$\mathbf{y}_t$ is an $m \times 1$ vector of dependent variables;

$\mathbf{C}$ is an $m \times k$ matrix of parameters;

$\mathbf{x}_t$ is a $k \times 1$ vector of independent variables, which may contain lags of $\mathbf{y}_t$;

$\mathbf{H}_t^{1/2}$ is the Cholesky factor of the time-varying conditional covariance matrix $\mathbf{H}_t$;

$\boldsymbol{\nu}_t$ is an $m \times 1$ vector of independent and identically distributed innovations;

$\mathbf{D}_t$ is a diagonal matrix of conditional variances,

$$\mathbf{D}_t = \begin{pmatrix} \sigma^2_{1,t} & 0 & \cdots & 0 \\ 0 & \sigma^2_{2,t} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma^2_{m,t} \end{pmatrix}$$

in which each $\sigma^2_{i,t}$ evolves according to a univariate GARCH model of the form

$$\sigma^2_{i,t} = s_i + \sum_{j=1}^{p_i} \alpha_j \epsilon^2_{i,t-j} + \sum_{j=1}^{q_i} \beta_j \sigma^2_{i,t-j}$$

by default, or

$$\sigma^2_{i,t} = \exp(\boldsymbol{\gamma}_i \mathbf{z}_{i,t}) + \sum_{j=1}^{p_i} \alpha_j \epsilon^2_{i,t-j} + \sum_{j=1}^{q_i} \beta_j \sigma^2_{i,t-j}$$

when the `het()` option is specified, where $\boldsymbol{\gamma}_t$ is a $1 \times p$ vector of parameters, $\mathbf{z}_i$ is a $p \times 1$ vector of independent variables including a constant term, the α_j's are ARCH parameters, and the β_j's are GARCH parameters;

$\mathbf{R}_t$ is a matrix of conditional correlations,

$$\mathbf{R}_t = \begin{pmatrix} 1 & \rho_{12,t} & \cdots & \rho_{1m,t} \\ \rho_{12,t} & 1 & \cdots & \rho_{2m,t} \\ \vdots & \vdots & \ddots & \vdots \\ \rho_{1m,t} & \rho_{2m,t} & \cdots & 1 \end{pmatrix}$$

$\mathbf{R}$ is the matrix of means to which the dynamic process in (1) reverts;

$\boldsymbol{\Psi}_t$ is the rolling estimator of the correlation matrix of $\widetilde{\boldsymbol{\epsilon}}_t$, which uses the previous $m + 1$ observations; and

λ_1 and λ_2 are parameters that govern the dynamics of conditional correlations. λ_1 and λ_2 are nonnegative and satisfy $0 \leq \lambda_1 + \lambda_2 < 1$.

To differentiate this model from Engle (2002), Tse and Tsui (2002) call their model a VCC MGARCH model.

❑

Some examples

▷ Example 1: Model with common covariates

We have daily data on the stock returns of three car manufacturers—Toyota, Nissan, and Honda, from January 2, 2003, to December 31, 2010—in the variables `toyota`, `nissan`, and `honda`. We model the conditional means of the returns as a first-order vector autoregressive process and the conditional covariances as a VCC MGARCH process in which the variance of each disturbance term follows a GARCH(1,1) process.

```
. use http://www.stata-press.com/data/r14/stocks
(Data from Yahoo! Finance)
. mgarch vcc (toyota nissan honda = L.toyota L.nissan L.honda, noconstant),
> arch(1) garch(1)
Calculating starting values....
Optimizing log likelihood
(setting technique to bhhh)
Iteration 0:   log likelihood =    16901.2
Iteration 1:   log likelihood =  17028.644
Iteration 2:   log likelihood =  17145.905
Iteration 3:   log likelihood =  17251.485
Iteration 4:   log likelihood =  17306.115
Iteration 5:   log likelihood =   17332.59
Iteration 6:   log likelihood =  17353.617
Iteration 7:   log likelihood =   17374.86
Iteration 8:   log likelihood =  17398.526
Iteration 9:   log likelihood =  17418.748
(switching technique to nr)
Iteration 10:  log likelihood =  17442.552
Iteration 11:  log likelihood =   17455.64
Iteration 12:  log likelihood =  17463.593
Iteration 13:  log likelihood =  17463.922
Iteration 14:  log likelihood =  17463.925
Iteration 15:  log likelihood =  17463.925
Refining estimates
Iteration 0:   log likelihood =  17463.925
Iteration 1:   log likelihood =  17463.925
Varying conditional correlation MGARCH model
Sample: 1 - 2015                                Number of obs   =      2,014
Distribution: Gaussian                          Wald chi2(9)    =      17.67
Log likelihood =  17463.92                      Prob > chi2     =     0.0392
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
toyota						
toyota						
L1.	-.0565645	.0335696	-1.68	0.092	-.1223597	.0092307
nissan						
L1.	.0248101	.0252701	0.98	0.326	-.0247184	.0743385
honda						
L1.	.0035836	.0298895	0.12	0.905	-.0549986	.0621659
ARCH_toyota						
arch						
L1.	.0602807	.0086799	6.94	0.000	.0432684	.0772929
garch						
L1.	.922469	.0110316	83.62	0.000	.9008474	.9440906
_cons	4.38e-06	1.12e-06	3.91	0.000	2.18e-06	6.58e-06

nissan						
toyota						
L1.	-.0196399	.0387112	-0.51	0.612	-.0955124	.0562326
nissan						
L1.	-.0306663	.031051	-0.99	0.323	-.091525	.0301925
honda						
L1.	.038315	.0354691	1.08	0.280	-.0312031	.1078331
ARCH_nissan						
arch						
L1.	.0774228	.0119642	6.47	0.000	.0539733	.1008723
garch						
L1.	.9076856	.0139339	65.14	0.000	.8803756	.9349955
_cons	6.20e-06	1.70e-06	3.65	0.000	2.87e-06	9.53e-06
honda						
toyota						
L1.	-.0358293	.0340492	-1.05	0.293	-.1025645	.030906
nissan						
L1.	.0544071	.0276156	1.97	0.049	.0002815	.1085327
honda						
L1.	-.0424383	.0326249	-1.30	0.193	-.106382	.0215054
ARCH_honda						
arch						
L1.	.0458673	.0072714	6.31	0.000	.0316157	.0601189
garch						
L1.	.9369252	.0101755	92.08	0.000	.9169815	.9568689
_cons	4.99e-06	1.29e-06	3.85	0.000	2.45e-06	7.52e-06
corr(toyota, nissan)	.6643028	.0151086	43.97	0.000	.6346905	.6939151
corr(toyota, honda)	.7302093	.0126361	57.79	0.000	.705443	.7549755
corr(nissan, honda)	.6347321	.0159738	39.74	0.000	.603424	.6660401
Adjustment						
lambda1	.0277374	.0086942	3.19	0.001	.010697	.0447778
lambda2	.8255525	.0755881	10.92	0.000	.6774025	.9737026

The output has three parts: an iteration log, a header, and an output table.

The iteration log has three parts: the dots from the search for initial values, the iteration log from optimizing the log likelihood, and the iteration log from the refining step. A detailed discussion of the optimization methods is in *Methods and formulas*.

The header describes the estimation sample and reports a Wald test against the null hypothesis that all the coefficients on the independent variables in the mean equations are zero. Here the null hypothesis is rejected at the 5% level.

The output table first presents results for the mean or variance parameters used to model each dependent variable. Subsequently, the output table presents results for the parameters in $\mathbf{R}$. For example, the estimate of the mean of the process that associates Toyota and Nissan is 0.66. Finally, the output table presents results for the adjustment parameters λ_1 and λ_2. In the example at hand, the estimates for both λ_1 and λ_2 are statistically significant.

The VCC MGARCH model reduces to the CCC MGARCH model when $\lambda_1 = \lambda_2 = 0$. The output below shows that a Wald test rejects the null hypothesis that $\lambda_1 = \lambda_2 = 0$ at all conventional levels.

```
. test _b[Adjustment:lambda1] = _b[Adjustment:lambda2] = 0
 ( 1)  [Adjustment]lambda1 - [Adjustment]lambda2 = 0
 ( 2)  [Adjustment]lambda1 = 0
           chi2(  2) =  482.80
         Prob > chi2 =    0.0000
```

These results indicate that the assumption of time-invariant conditional correlations maintained in the CCC MGARCH model is too restrictive for these data.

◁

▷ Example 2: Model with covariates that differ by equation

We improve the previous example by removing the insignificant parameters from the model. To accomplish that, we specify the `honda` equation separately from the `toyota` and `nissan` equations:

```
. mgarch vcc (toyota nissan = , noconstant) (honda = L.nissan, noconstant),
> arch(1) garch(1)
Calculating starting values....
Optimizing log likelihood
(setting technique to bhhh)
Iteration 0:   log likelihood =   16889.43
Iteration 1:   log likelihood =  17002.567
Iteration 2:   log likelihood =  17134.525
Iteration 3:   log likelihood =  17233.192
Iteration 4:   log likelihood =  17295.342
Iteration 5:   log likelihood =  17326.347
Iteration 6:   log likelihood =  17348.063
Iteration 7:   log likelihood =  17363.988
Iteration 8:   log likelihood =  17387.216
Iteration 9:   log likelihood =  17404.734
(switching technique to nr)
Iteration 10:  log likelihood =  17438.432  (not concave)
Iteration 11:  log likelihood =  17450.002
Iteration 12:  log likelihood =  17455.443
Iteration 13:  log likelihood =  17455.971
Iteration 14:  log likelihood =   17455.98
Iteration 15:  log likelihood =   17455.98
Refining estimates
Iteration 0:   log likelihood =   17455.98
Iteration 1:   log likelihood =   17455.98  (backed up)
```

```
Varying conditional correlation MGARCH model
Sample: 1 - 2015                                  Number of obs   =      2,014
Distribution: Gaussian                            Wald chi2(1)    =       1.62
Log likelihood =  17455.98                        Prob > chi2     =     0.2032
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
ARCH_toyota						
arch						
L1.	.0609064	.0087784	6.94	0.000	.0437011	.0781117
garch						
L1.	.921703	.0111493	82.67	0.000	.8998508	.9435552
_cons	4.42e-06	1.13e-06	3.91	0.000	2.20e-06	6.64e-06
ARCH_nissan						
arch						
L1.	.0806598	.0123529	6.53	0.000	.0564486	.104871
garch						
L1.	.9035239	.014421	62.65	0.000	.8752592	.9317886
_cons	6.61e-06	1.79e-06	3.70	0.000	3.11e-06	.0000101
honda						
nissan						
L1.	.0175565	.0137982	1.27	0.203	-.0094874	.0446005
ARCH_honda						
arch						
L1.	.0461398	.0073048	6.32	0.000	.0318225	.060457
garch						
L1.	.9366096	.0102021	91.81	0.000	.9166139	.9566053
_cons	5.03e-06	1.31e-06	3.85	0.000	2.47e-06	7.59e-06
corr(toyota, nissan)	.6635251	.0150293	44.15	0.000	.6340682	.692982
corr(toyota, honda)	.7299703	.0124828	58.48	0.000	.7055045	.754436
corr(nissan, honda)	.6338207	.0158681	39.94	0.000	.6027198	.6649217
Adjustment						
lambda1	.0285319	.0092448	3.09	0.002	.0104124	.0466514
lambda2	.8113923	.0854955	9.49	0.000	.6438242	.9789604

It turns out that the coefficient on `L1.nissan` in the `honda` equation is now statistically insignificant. We could further improve the model by removing `L1.nissan` from the model.

There is no mean equation for Toyota or Nissan. In [TS] **mgarch vcc postestimation**, we discuss prediction from models without covariates.

◁

Example 3: Model with constraints

Here we fit a bivariate VCC MGARCH model for the Toyota and Nissan shares. We believe that the shares of these car manufacturers follow the same process, so we impose the constraints that the ARCH coefficients are the same for the two companies and that the GARCH coefficients are also the same.

```
. constraint 1 _b[ARCH_toyota:L.arch] = _b[ARCH_nissan:L.arch]
. constraint 2 _b[ARCH_toyota:L.garch] = _b[ARCH_nissan:L.garch]
. mgarch vcc (toyota nissan = , noconstant), arch(1) garch(1) constraints(1 2)
Calculating starting values....
Optimizing log likelihood
(setting technique to bhhh)
Iteration 0:   log likelihood =  10326.298
Iteration 1:   log likelihood =   10680.73
Iteration 2:   log likelihood =  10881.388
Iteration 3:   log likelihood =  11043.345
Iteration 4:   log likelihood =  11122.459
Iteration 5:   log likelihood =  11202.411
Iteration 6:   log likelihood =  11253.657
Iteration 7:   log likelihood =  11276.325
Iteration 8:   log likelihood =  11279.823
Iteration 9:   log likelihood =  11281.704
(switching technique to nr)
Iteration 10:  log likelihood =  11282.313
Iteration 11:  log likelihood =   11282.46
Iteration 12:  log likelihood =  11282.461
```

```
Refining estimates
Iteration 0:   log likelihood =  11282.461
Iteration 1:   log likelihood =  11282.461  (backed up)
Varying conditional correlation MGARCH model
Sample: 1 - 2015                                Number of obs   =      2,015
Distribution: Gaussian                          Wald chi2(.)    =          .
Log likelihood =  11282.46                      Prob > chi2     =          .
 ( 1)  [ARCH_toyota]L.arch - [ARCH_nissan]L.arch = 0
 ( 2)  [ARCH_toyota]L.garch - [ARCH_nissan]L.garch = 0
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
ARCH_toyota						
arch						
L1.	.0797459	.0101634	7.85	0.000	.059826	.0996659
garch						
L1.	.9063808	.0118211	76.67	0.000	.883212	.9295497
_cons	4.24e-06	1.10e-06	3.85	0.000	2.08e-06	6.40e-06
ARCH_nissan						
arch						
L1.	.0797459	.0101634	7.85	0.000	.059826	.0996659
garch						
L1.	.9063808	.0118211	76.67	0.000	.883212	.9295497
_cons	5.91e-06	1.47e-06	4.03	0.000	3.03e-06	8.79e-06
corr(toyota, nissan)	.6720056	.0162585	41.33	0.000	.6401394	.7038718
Adjustment						
lambda1	.0343012	.0128097	2.68	0.007	.0091945	.0594078
lambda2	.7945548	.101067	7.86	0.000	.596467	.9926425

We could test our constraints by fitting the unconstrained model and performing a likelihood-ratio test. The results indicate that the restricted model is preferable.

◁

▷ Example 4: Model with a GARCH term

In this example, we have data on fictional stock returns for the Acme and Anvil corporations, and we believe that the movement of the two stocks is governed by different processes. We specify one ARCH and one GARCH term for the conditional variance equation for Acme and two ARCH terms for the conditional variance equation for Anvil. In addition, we include the lagged value of the stock return for Apex, the main subsidiary of Anvil corporation, in the variance equation of Anvil. For Acme, we have data on the changes in an index of futures prices of products related to those produced by Acme in `afrelated`. For Anvil, we have data on the changes in an index of futures prices of inputs used by Anvil in `afinputs`.

```
. use http://www.stata-press.com/data/r14/acmeh
. mgarch vcc (acme = afrelated, noconstant arch(1) garch(1))
> (anvil = afinputs, arch(1/2) het(L.apex))
Calculating starting values....
```

```
Optimizing log likelihood
(setting technique to bhhh)
Iteration 0:   log likelihood = -13252.793
Iteration 1:   log likelihood = -12859.124
Iteration 2:   log likelihood =  -12522.14
Iteration 3:   log likelihood = -12406.487
Iteration 4:   log likelihood = -12304.275
Iteration 5:   log likelihood = -12273.103
Iteration 6:   log likelihood = -12256.104
Iteration 7:   log likelihood =  -12254.55
Iteration 8:   log likelihood = -12254.482
Iteration 9:   log likelihood = -12254.478
(switching technique to nr)
Iteration 10:  log likelihood = -12254.478
Iteration 11:  log likelihood = -12254.478
Refining estimates
Iteration 0:   log likelihood = -12254.478
Iteration 1:   log likelihood = -12254.478
Varying conditional correlation MGARCH model
Sample: 1 - 2500                                Number of obs   =      2,499
Distribution: Gaussian                          Wald chi2(2)    =    5226.19
Log likelihood = -12254.48                      Prob > chi2     =     0.0000
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
acme						
afrelated	.9672465	.0510066	18.96	0.000	.8672753	1.067218
ARCH_acme						
arch						
L1.	.0949142	.0147302	6.44	0.000	.0660435	.1237849
garch						
L1.	.7689442	.038885	19.77	0.000	.6927309	.8451574
_cons	2.129468	.464916	4.58	0.000	1.218249	3.040687
anvil						
afinputs	-1.018629	.0145027	-70.24	0.000	-1.047053	-.9902037
_cons	.1015986	.0177952	5.71	0.000	.0667205	.1364766
ARCH_anvil						
arch						
L1.	.4990272	.0243531	20.49	0.000	.4512959	.5467584
L2.	.2839812	.0181966	15.61	0.000	.2483165	.3196459
apex						
L1.	1.897144	.0558791	33.95	0.000	1.787623	2.006665
_cons	.0682724	.0662257	1.03	0.303	-.0615276	.1980724
corr(acme, anvil)	-.6574256	.0294259	-22.34	0.000	-.7150994	-.5997518
Adjustment						
lambda1	.2375029	.0179114	13.26	0.000	.2023971	.2726086
lambda2	.6492072	.0254493	25.51	0.000	.5993274	.6990869

The results indicate that increases in the futures prices for related products lead to higher returns on the Acme stock, and increased input prices lead to lower returns on the Anvil stock. In the conditional variance equation for Anvil, the coefficient on `L1.apex` is positive and significant, which indicates that an increase in the return on the Apex stock leads to more variability in the return on the Anvil stock.

◁

Stored results

`mgarch vcc` stores the following in `e()`:

Scalars

`e(N)`	number of observations
`e(k)`	number of parameters
`e(k_aux)`	number of auxiliary parameters
`e(k_extra)`	number of extra estimates added to `_b`
`e(k_eq)`	number of equations in `e(b)`
`e(k_dv)`	number of dependent variables
`e(df_m)`	model degrees of freedom
`e(ll)`	log likelihood
`e(chi2)`	χ^2
`e(p)`	significance
`e(estdf)`	`1` if distribution parameter was estimated, `0` otherwise
`e(usr)`	user-provided distribution parameter
`e(tmin)`	minimum time in sample
`e(tmax)`	maximum time in sample
`e(N_gaps)`	number of gaps
`e(rank)`	rank of `e(V)`
`e(ic)`	number of iterations
`e(rc)`	return code
`e(converged)`	`1` if converged, `0` otherwise

Macros

`e(cmd)`	`mgarch`
`e(model)`	`vcc`
`e(cmdline)`	command as typed
`e(depvar)`	names of dependent variables
`e(covariates)`	list of covariates
`e(dv_eqs)`	dependent variables with mean equations
`e(indeps)`	independent variables in each equation
`e(tvar)`	time variable
`e(title)`	title in estimation output
`e(chi2type)`	`Wald`; type of model χ^2 test
`e(vce)`	*vcetype* specified in `vce()`
`e(vcetype)`	title used to label Std. Err.
`e(tmins)`	formatted minimum time
`e(tmaxs)`	formatted maximum time
`e(dist)`	distribution for error term: `gaussian` or `t`
`e(arch)`	specified ARCH terms
`e(garch)`	specified GARCH terms
`e(technique)`	maximization technique
`e(properties)`	`b V`
`e(estat_cmd)`	program used to implement `estat`
`e(predict)`	program used to implement `predict`
`e(marginsok)`	predictions allowed by `margins`
`e(marginsnotok)`	predictions disallowed by `margins`
`e(marginsdefault)`	default `predict()` specification for `margins`
`e(asbalanced)`	factor variables `fvset` as `asbalanced`
`e(asobserved)`	factor variables `fvset` as `asobserved`

Matrices

e(b)	coefficient vector
e(Cns)	constraints matrix
e(ilog)	iteration log (up to 20 iterations)
e(gradient)	gradient vector
e(hessian)	Hessian matrix
e(V)	variance–covariance matrix of the estimators
e(pinfo)	parameter information, used by predict

Functions

e(sample)	marks estimation sample

Methods and formulas

mgarch vcc estimates the parameters of the varying conditional correlation MGARCH model by maximum likelihood. The log-likelihood function based on the multivariate normal distribution for observation t is

$$l_t = -0.5m\log(2\pi) - 0.5\log\{\det(\mathbf{R}_t)\} - \log\left\{\det\left(\mathbf{D}_t^{1/2}\right)\right\} - 0.5\widetilde{\boldsymbol{\epsilon}}_t\mathbf{R}_t^{-1}\widetilde{\boldsymbol{\epsilon}}_t'$$

where $\widetilde{\boldsymbol{\epsilon}}_t = \mathbf{D}_t^{-1/2}\boldsymbol{\epsilon}_t$ is an $m \times 1$ vector of standardized residuals, $\boldsymbol{\epsilon}_t = \mathbf{y}_t - \mathbf{C}\mathbf{x}_t$. The log-likelihood function is $\sum_{t=1}^{T} l_t$.

If we assume that $\boldsymbol{\nu}_t$ follow a multivariate t distribution with degrees of freedom (df) greater than 2, then the log-likelihood function for observation t is

$$\begin{aligned} l_t = {} & \log\Gamma\left(\frac{\mathrm{df}+m}{2}\right) - \log\Gamma\left(\frac{\mathrm{df}}{2}\right) - \frac{m}{2}\log\{(\mathrm{df}-2)\pi\} \\ & - 0.5\log\{\det(\mathbf{R}_t)\} - \log\left\{\det\left(\mathbf{D}_t^{1/2}\right)\right\} - \frac{\mathrm{df}+m}{2}\log\left(1 + \frac{\widetilde{\boldsymbol{\epsilon}}_t\mathbf{R}_t^{-1}\widetilde{\boldsymbol{\epsilon}}_t'}{\mathrm{df}-2}\right) \end{aligned}$$

The starting values for the parameters in the mean equations and the initial residuals $\widehat{\boldsymbol{\epsilon}}_t$ are obtained by least-squares regression. The starting values for the parameters in the variance equations are obtained by a procedure proposed by Gourieroux and Monfort (1997, sec. 6.2.2). The starting values for the parameters in $\mathbf{R}$ are calculated from the standardized residuals $\widetilde{\boldsymbol{\epsilon}}_t$. Given the starting values for the mean and variance equations, the starting values for the parameters λ_1 and λ_2 are obtained from a grid search performed on the log likelihood.

The initial optimization step is performed in the unconstrained space. Once the maximum is found, we impose the constraints $\lambda_1 \geq 0$, $\lambda_2 \geq 0$, and $0 \leq \lambda_1 + \lambda_2 < 1$, and maximize the log likelihood in the constrained space. This step is reported in the iteration log as the refining step.

GARCH estimators require initial values that can be plugged in for $\boldsymbol{\epsilon}_{t-i}\boldsymbol{\epsilon}_{t-i}'$ and $\mathbf{H}_{t-j}$ when $t-i<1$ and $t-j<1$. mgarch vcc substitutes an estimator of the unconditional covariance of the disturbances

$$\widehat{\boldsymbol{\Sigma}} = T^{-1}\sum_{t=1}^{T}\widehat{\widetilde{\boldsymbol{\epsilon}}}_t\widehat{\widetilde{\boldsymbol{\epsilon}}}_t' \tag{2}$$

for $\boldsymbol{\epsilon}_{t-i}\boldsymbol{\epsilon}_{t-i}'$ when $t-i<1$ and for $\mathbf{H}_{t-j}$ when $t-j<1$, where $\widehat{\widehat{\boldsymbol{\epsilon}}}_t$ is the vector of residuals calculated using the estimated parameters.

`mgarch vcc` uses numerical derivatives in maximizing the log-likelihood function.

References

Engle, R. F. 2002. Dynamic conditional correlation: A simple class of multivariate generalized autoregressive conditional heteroskedasticity models. *Journal of Business & Economic Statistics* 20: 339–350.

Gourieroux, C. S., and A. Monfort. 1997. *Time Series and Dynamic Models*. Trans. ed. G. M. Gallo. Cambridge: Cambridge University Press.

Tse, Y. K., and A. K. C. Tsui. 2002. A multivariate generalized autoregressive conditional heteroscedasticity model with time-varying correlations. *Journal of Business & Economic Statistics* 20: 351–362.

Also see

[TS] **mgarch vcc postestimation** — Postestimation tools for mgarch vcc

[TS] **mgarch** — Multivariate GARCH models

[TS] **tsset** — Declare data to be time-series data

[TS] **arch** — Autoregressive conditional heteroskedasticity (ARCH) family of estimators

[TS] **var** — Vector autoregressive models

[U] **20 Estimation and postestimation commands**